Implementation and Performance of AES-NI in CyaSSL

Embedded SSL

In 2010, Intel introduced the 32nm Intel® microarchitecture code name Westmere. With this introduction, Intel announced support for a new set of hardware-based Advanced Encryption Standard (AES) instructions. These instructions, named "AES-NI" (or AES "New Instructions") are composed of six individual instructions in total, offering hardware accelerated AES routines to both consumers and developers worldwide.

Contents

Introduction

In 2010, Intel introduced the 32nm Intel® microarchitecture code name Westmere. With this introduction, Intel announced support for a new set of hardware-based Advanced Encryption Standard (AES) instructions. These instructions, named Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI) are composed of six individual instructions, offering hardware accelerated AES routines to both consumers and developers worldwide.

This paper, authored by yaSSL, provides a brief overview of the Intel AES-NI instructions and demonstrates the performance gains realized when yaSSL used Intel AES-NI in place of a more traditional software-only based AES implementation. The CyaSSL embedded SSL library* developed by yaSSL, is used as a test bed to perform the comparison, as it can be built with either traditional AES or AES-NI support at compile time. As a secondary goal to demonstrating Intel AES-NI performance, this paper explains how to determine if a pre-built SSL library (static or shared) offers built-in support for the Intel Advanced Encryption Standard New Instructions.

Audience

This paper is intended for software developers and systems administrators. It can be used to understand the performance benefit of using new instructions in place of a standard software-based AES implementation and will introduce the benefits of using a small, lightweight implementation of SSL such as CyaSSL on platforms ranging from embedded devices to enterprise cloud servers with millions of active SSL connections.

CyaSSL Embedded SSL Library

The CyaSSL embedded SSL library is a lightweight SSL/TLS library written in the C programming language. It is mainly targeted at embedded and RTOS environments –

primarily because of its small size, speed, and feature set – but is commonly used to secure desktop and enterprise environments as well because if its scalability, royalty-free pricing model, and excellent cross-platform support.

CyaSSL supports industry SSL standards up to the current TLS 1.2 level, offers very small per-session runtime memory usage, progressive ciphers, several abstraction layers, and is typically up to 20 times smaller than OpenSSL. It offers users a full list of features including an OpenSSL compatibility layer, DTLS, OCSP, and CRL support, SSL inspection capabilities, x509 certificate generation, and has support for client authentication. Footprint sizes are typically 30-100kB for a fully-featured TLS 1.2 compliant client and server, depending on build options and operating environment, and runtime memory usage is between 3-36kB per SSL session. With memory usages kept to a minimum, CyaSSL is perfect for adding secure communication to resource constrained embedded devices. It is also ideal for use in large-scale enterprise environments where it can help reduce the number of resources needed (translating to fewer machines) to handle high loads of concurrent SSL connections.

As a leader in the embedded security industry, yaSSL provides customers with open source products that are feature-rich, highly portable, optimized for resource-constrained environments, and backed by a very dedicated and responsive support and development team. yaSSL products are designed to offer optimal embedded and enterprise performance, rapid integration into existing applications and platforms, the ability to leverage hardware cryptography solutions such as Intel's AES-NI, and support for the most current standards. All products are designed for ease-of-use with clean and simple APIs and are backed by a company with a proven track record in the software and communication security industries.

CyaSSL is dual licensed under both the GPLv2 (http://www.gnu.org/licenses/gpl-2.0.html) as well as a standard commercial license. GPLv2-licensed versions of CyaSSL, as well as documentation and examples, may be downloaded directly from the yaSSL website (www.yassl.com).

Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI) Overview

Intel AES-NI offers full hardware support for data encryption and decryption using the Advanced Encryption Standard. Four of the instructions support AES encryption and decryption while two support AES key expansion.

AES is a symmetric key algorithm, meaning that the same key is used for both encryption and decryption. It was first announced by the National Institute of Standards (NIST) in FIPS PUB 197, then went through a standardization and algorithm selection process, finally becoming a United States federal government standard in 2002.

Intel AES-NI (as well as standard AES itself) has the flexibility to support key lengths of 128, 192, and 256 bits by processing the data block of plaintext in 10, 12, or 14 rounds of redefined transformations respectively. Each round performs a number of operations on the input block, which is then fed into the next round as input. Because the instructions are hardware-based, they offer a significant increase in performance compared to current software implementations of AES.

Beyond improving performance, these new AES instructions provide important security benefits. By running in data-independent time and not using tables, they help in eliminating the major timing and cache-based attacks that threaten table-based software implementations of AES. In addition, these instructions make AES simple to implement, with reduced code size – thus helping reduce the risk of the inadvertent introduction of security flaws which lead to difficult-to-detect side channel attacks.

The six Intel AES-NI instructions used to accelerate symmetric block encryption/decryption of data are listed below.

- **AESENC** and **AESENCLAST** - perform AES encryption, with AESENC performing a single round of encryption and AESENCLAST performing the final round of encryption on the data block. Each instruction groups several operations into a single instruction, including *ShiftRows*, *SubBytes*, and *MixColumns*.

- **AESDEC** and **AESDECLAST** – perform AES decryption, with AESDEC performing a single round of decryption and AESDECLAST performing the final round of decryption on the data block. These instructions encapsulate the *InvShiftRows*, *InvSubBytes*, and *InvMixColumns* operations.
- **AESIMC** – easily allows round keys to be converted into a usable form for decryption using the Equivalent Inverse Cipher.
- **AESKEYGENASSIST** – used for the generation of round keys for AES encryption/decryption routines.

For a complete description of the AES-NI instructions, please refer to section 12.13 in the Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C (http://download.intel.com/products/processor/manual/325462.pdf).

System Setup and Configuration

The benchmarks and comparisons in this document were generated using the system and hardware setup as explained in Table 1, below.

All software tests in this paper revolve around CyaSSL*. SSL is a very crypto-intensive protocol. As such, CyaSSL contains the underlying CTaoCrypt cryptography library that provides cryptography operations supporting SSL/TLS. CTaoCrypt offers both a traditional software-based AES implementation as well as the ability to leverage Intel's AES-NI.

Included in the CyaSSL download is a CTaoCrypt benchmark utility. This utility is the tool used to measure AES performance on the test system. After downloading and building CyaSSL, the benchmark utility is located at "<cyassl_root>/ctaocrypt/benchmark/benchmark", under the main CyaSSL root directory location.

Table 1: Hardware Components

Component	Details
Test Hardware	Intel Core i7, 2.2 GHz, 8 GB RAM, 500 GB HDD
Operating System	Ubuntu* Server 12.04 LTS 64 bit
Library / Application Software	CyaSSL 2.3.0
Benchmark Tools	CTaoCrypt benchmark utility

Software Setup

CyaSSL can be downloaded from the yaSSL website under the GPLv2 license. For commercial licenses, please contact yaSSL directly.

http://yassl.com/yaSSL/download/downloadForm.php

There are several steps required to build the CyaSSL library and supporting test software. By default, CyaSSL uses the autoconf system to configure and build the library. For the following tests a baseline build was compiled from the terminal as follows. This configuration will build CyaSSL with the default software-based AES implementation.

```
% cd cyassl-2.3.0
% ./configure
% make
% make install
```

The baseline test suite is executed as follows and should confirm a valid build of the application and library. The test suite runs fifteen cryptography tests in addition to client/server SSL tests, unit tests, and cipher suite tests. All tests should pass on a successfully compiled CyaSSL package.

```
% make test
```

To build CyaSSL with the new AES-NI instructions, repeat the above steps but modify the configure command to enable Intel AES-NI support by adding the "—enable-aesni" configure option.

```
% cd cyassl-2.3.0
% make clean
% ./configure --enable-aesni
% make
% make install
```

Determination of library support for Intel® AES-NI

For many developers who are using third party SSL libraries, the developer may not know if the library explicitly supports AES-NI. Below is a method for quickly determining if Intel AES-NI is enabled in a compiled SSL/TLS library. In the case of the CyaSSL library built in the previous section, we can see that CyaSSL has been built with AES-NI support from the disassembly and grep steps below.

```
% cd cyassl-2.3.0/src/.libs
% objdump -D  libcyassl.so | grep AESENC
% objdump -D  libcyassl.so | grep AESDEC
% objdump -D  libcyassl.so | grep AESDECLAST
% objdump -D  libcyassl.so | grep AESENCLAST
% objdump -D  libcyassl.so | grep AESIMC
```

The above commands are issued on a shared library (.so extension). If an SSL library has been built as a static library (.a extension), the steps to check for AES-NI implementation would be the same, for example:

```
% objdump -D  libcyassl.a | grep AESENC
```

Performance Tests

To test the AES performance improvement gained in CyaSSL when enabling Intel AES-NI, several tests were run using the CTaoCrypt benchmark application. To run the benchmark application, issue the following command from the CyaSSL root directory:

```
% ./ctaocrypt/benchmark/benchmark
```

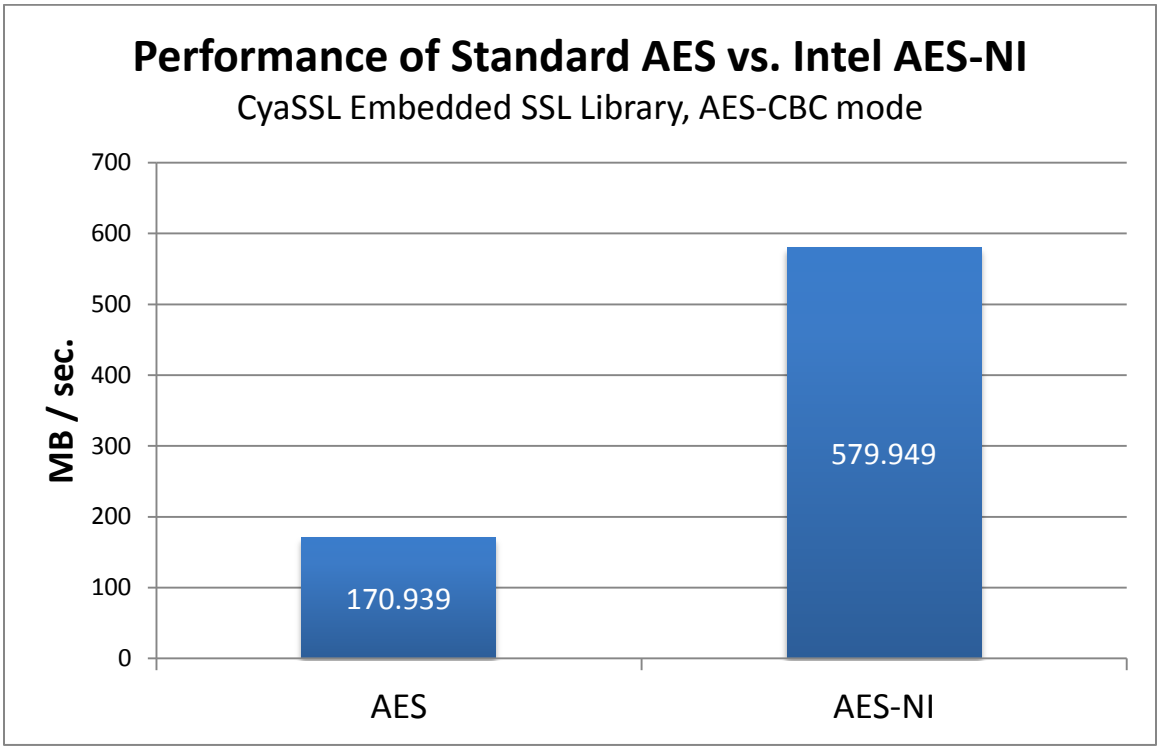
Typical benchmark output looks similar to:

```
AES      5 megs took 0.029 seconds, 172.82 MB/s
ARC4     5 megs took 0.020 seconds, 246.03 MB/s
RABBIT   5 megs took 0.014 seconds, 368.00 MB/s
3DES     5 megs took 0.236 seconds,  21.16 MB/s

MD5      5 megs took 0.014 seconds, 368.70 MB/s
SHA      5 megs took 0.022 seconds, 226.42 MB/s
SHA-256  5 megs took 0.045 seconds, 110.39 MB/s

RSA 2048 encryption took   0.73 milliseconds, avg over 100 iterations
RSA 2048 decryption took   4.02 milliseconds, avg over 100 iterations
DH  2048 key generation    1.10 milliseconds, avg over 100 iterations
DH  2048 key agreement     1.75 milliseconds, avg over 100 iterations
```

For comparing AES performance, the only test focused on is the AES test in the benchmark tests (shown in bold above). The CTaoCrypt benchmark application was run 10 consecutive times for both standard AES and AES-NI. The results of these tests were averaged together for AES and AES-NI and the results have been collected in Figure 1, below.

The AES tests performed by the CTaoCrypt benchmark application utilized AES in CBC mode, doing straight AES encryption. To most accurately measure AES performance, the individual algorithm performance was measured separately from SSL/TLS communication.

Figure 1: Performance of AES vs. Intel AES-NI in CyaSSL



Looking at the results in Figure 1, we see that by building CyaSSL with Intel AES-NI support, AES-CBC performance is over 3.3 times faster than using traditional software-based AES when doing direct encryption.

Conclusion

By enabling Intel AES-NI in a given application or library, users are able to see a substantial performance increase in AES operations. The performance increase gained when using Intel AES-NI combined with the reduced risk of security attacks makes an arguable point to enable Intel AES-NI on production environments of all types. As the CyaSSL embedded SSL library provides support for both traditional software-based AES as well as Intel's AES-NI, and offers a very easy to work with license model, users are able to effortlessly test the potential performance increase on their platform with CyaSSL. With the correct platform and configuration, users should be able to see a significant performance increase when using AES-NI over standard AES.

Terminology and Reference

Term	Description
AES	Advanced Encryption Standard (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)
AES-NI	Advanced Encryption Standard New Instructions (http://download.intel.com/products/processor/manual/325462.pdf, Section 12.13)
SSL / TLS	Secure Sockets Layer / Transport Layer Security (http://tools.ietf.org/html/rfc5246)
CyaSSL	CyaSSL Embedded SSL Library (http://yassl.com/yaSSL/Products-cyassl.html)

About the Authors

Robert Chesebrough is a technical analyst in the Partner Experience team with Developer Relations Division, and is responsible for bringing new security related content to the Intel software developer community. He has been a contributing courseware developer and instructor for Intel Academic Community for over 8 years. Prior this role, Robert was a senior technical consulting engineer with the compiler marketing and technical support group in Intel's Software Products division. He authored the "Intel® Compiler Black-Belt Users Guide to undocumented switches". He holds a BS in Physics from the University of New Mexico and has been a software developer for the US Department of Energy, Sandia National Labs and Los Alamos National Labs beginning in the early 1980's and also in the in the aerospace industry at SBS technologies in the late 1990's. He is married and has two children.

Chris Conlon is a software developer at yaSSL. Holding a Bachelor of Science in Computer Science from Montana State University, he works to find a balance between outdoor adventures and computing. Chris enjoys continually learning and strives to bring new and helpful things to the technology community. He has focused on embedded security with yaSSL for the past 2 years, working with products including the CyaSSL embedded SSL library and the yaSSL Embedded Web Server.

Notices

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804